1. Alexey

Program Name: Alexey.java **Input File: alexey.dat**

Alexey's cousin Isidora recently shared with her a new concept she had learned in CS class called bit string flicking, which included operations like **left shift, right shift, left circle,** and **right circle** operations. Alexey was fascinated with this new idea and decided to do some further research on it, and discovered other operations that were used with bit string manipulation, like **NOT**, **AND**, **OR**, and **XOR**, which she knew worked with logic operations, but now she discovered could also work with binary strings.

She continued using Isidora's notation of using **RS** for Right Shift, **LS** for Left Shift, **RC** for Right Circle, and **LC** for Left Circle as the first part of the command structure, with a slight adaption of using a single space instead a dash, a value, another single space, and then the base ten integer value to be shifted or circulated. She added **N** for Not, **A** for AND, **O** for OR, and **X** for XOR, each preceding the operands, with single space separation.

As she had learned from Isidora, the shift command **RS 4 45** will take the value 45, convert it to the binary string **101101**, and then perform a **Right Shift 4**, which means the rightmost 4 bits are eliminated, leaving 10 in binary as the result, which is equivalent to the base ten value **2**. The command **LS 2 13** converts the value 13 to its binary equivalent of **1101**, and performs a **Left Shift 2** operation, which essentially appends two zeroes to the back of the binary string, resulting in **110100**, or **52** base ten.

The circle commands result in a same size binary string as the original, with a Right Circle taking the rightmost bits and circling them to the other side, and likewise for the Left Circle, taking the leftmost digits and circling them to the other side. For example, the command **RC 3 81** will take the binary value for 81, or **1010001** in binary, take the three right most digits, 001, and circle them to the front of the string resulting in **0011010,** which is **26** in decimal.

The four new commands would work like this.

- **N 23** would take the binary equivalent for 23, which is 10111, and "flip" all of the bits, resulting in 01000, or just 1000, which is equivalent to the decimal integer value **8**.
- The prefix style expression **A 23 14** would perform the bitwise AND operation on the two values 23 and 14, which in binary would be 10111 AND 1110, resulting in 00110, or **6**.
- The expression **O 23 14** is equivalent to 10111 OR 1110, which results in 11111, or **31**.
- The expression **X 23 14** is equivalent to 10111 XOR 1110, which has a result of 11001, or **25** in base ten.

Input: Several commands as described above, each on one line. It is guaranteed that the command will work within the length of the string, with all results greater than or equal to zero.

Output: The resulting base ten value for the command.

Sample input:
```
RS 4 45
LS 2 13
RC 3 81
N 23
A 23 14
O 23 14
X 23 14
```
Sample output:
```
2
52
26
8
6
31
25
```